

# Chapter 1

## Non-Player Characters in Multiuser Games

Massively multiuser, persistent, online virtual worlds are emerging as important platforms for multiuser computer games, social interaction, education, design, defence and commerce. In these virtual worlds, non-player characters use artificial intelligence to take on roles as storytellers, enemies, opponents, partners and facilitators. Non-player characters share their world with hundreds of thousands of creative, unpredictable, human controlled characters. As the complexity and functionality of multiuser virtual worlds increases, non-player characters are becoming an increasingly challenging application for artificial intelligence techniques [1]. The rise of multiuser games in particular has created a need for new kinds of artificial intelligence approaches that can produce characters with adaptive and complex behaviours for large-scale, dynamic game environments. Players are demanding more believable and intelligent non-player characters to enhance their gaming experience [2].

Motivated reinforcement learning is an emerging artificial intelligence technology that provides a theoretical and practical approach for developing adaptive characters for multiuser games. Motivated reinforcement learning has advantages over existing character control algorithms because it allows the development of non-player characters that can generate dynamic behaviour and adapt in time with an unpredictable, changing game environment. Motivated reinforcement learning uses computational models of human motivation – such as curiosity, interest and competence – to empower non-player characters to self-identify new tasks on which to focus their attention. The characters then use a reinforcement learning component to learn behavioural cycles to perform these tasks. The result is an agent model for non-player characters that are continually evolving new behaviours as a response to their experiences in their environment.

Motivated reinforcement learning transforms character design because it provides non-player characters with a mechanism for open-ended, online

adaptation of their own behaviour. In this book we aim to provide game programmers, and those with an interest in artificial intelligence, with the knowledge required to develop adaptable, intelligent non-player characters that can take on a life of their own in dynamic, multiuser virtual worlds.

## 1.1 Types of Multiuser Games

Computer games can be classified in a range of genres including action games, role-playing games, adventure games, strategy games, simulation games, sports games and racing games. Of these genres, action, role-playing, simulation and racing games have been starting points for multiuser games. A smaller subset again, including role-playing games and simulation games, has emerged as genres for games in persistent virtual worlds. The following sections introduce some distinguishing characteristics of these genres.

### 1.1.1 *Massively Multiplayer Online Role-Playing Games*

Massively multiplayer online role-playing games (MMORPGs) such as *World of Warcraft*, *Ultima Online*, *Everquest* and *Asheron's Call* are defined by a cast of non-player characters (NPCs) who act as enemies, partners and support characters to provide challenges, offer assistance and support the storyline. These characters exist in a persistent virtual world in which thousands of human players take on roles such as warriors, magicians and thieves and play and interact with NPCs and each other. Over time, the landscape of these worlds evolves and changes as players build their own houses or castles and craft items such as furniture, armour or weapons to personalise their dwellings or sell to other players. Unlike computer games played in non-persistent worlds, persistent game worlds offer months rather than hours of game play, which must be supported by NPCs. However, current technologies used to build non-player enemy, partner and support characters tend to constrain them to a set of fixed behaviours that cannot evolve in time with the world in which they dwell. The resulting behaviour of NPCs has been limited to looping animations with a few scripted action sequences triggered by a player's actions [3].

Sophisticated NPCs have the potential to enrich game worlds by providing opportunities for interesting interactions with players, making the game world more interactive and thus improving the believability of the game [4].

### 1.1.2 Multiuser Simulation Games

Simulation games such as *The Sims* are distinguished by characters that can respond to certain changes in their environment with new behaviours. Human players modify the circumstances that surround NPCs in order to influence the emergence of certain types of character behaviour for which points may be awarded or which trigger a new phase of the game. Existing simulation and organic simulation games are, however, limited by the set of changes that players may make to the game environment while the game is in progress.

In contrast, human users in current virtual worlds such as *Second Life* [5], *Active Worlds* [6] and *There* [7] can use open-ended modelling tools to create and modify world content. This sort of open-ended modelling is not available in many existing game worlds. The popularity and rapid growth in the user base of open-ended virtual worlds suggests the viability of a new generation of computer game situated in open-ended environments. A key challenge to be overcome in the development of such games, however, is the development of NPCs that can respond autonomously to the open-ended changes to their environment.

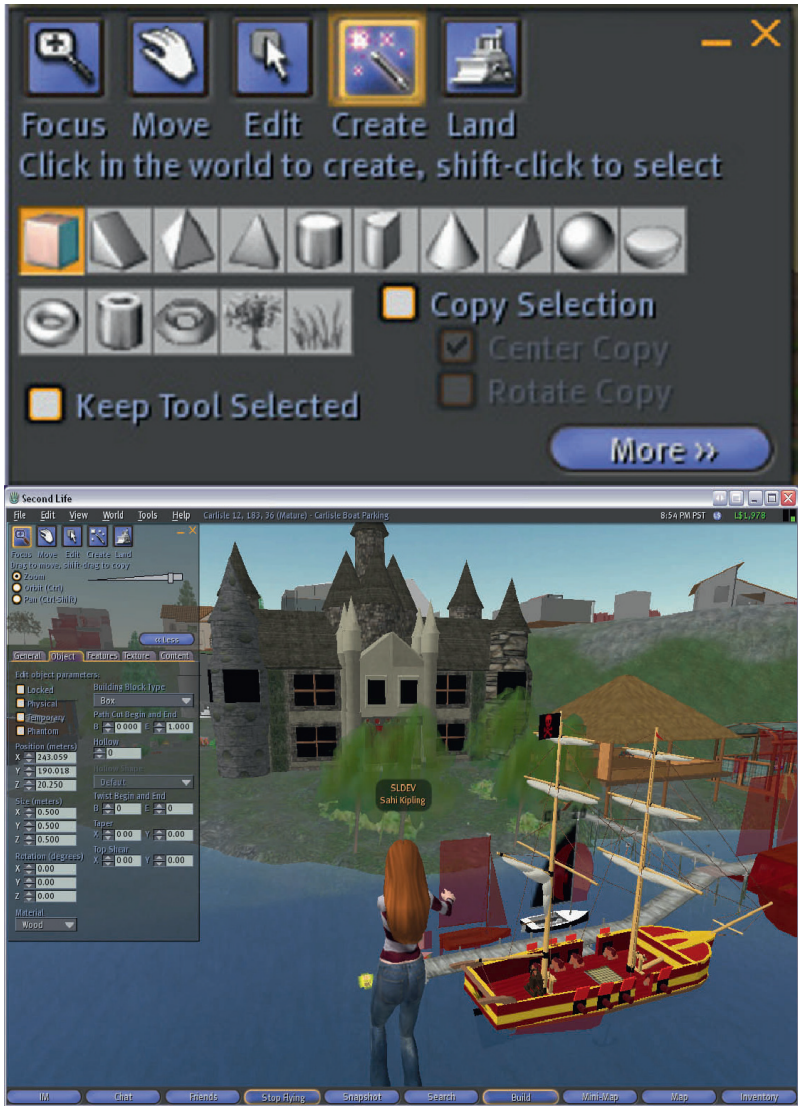
### 1.1.3 Open-Ended Virtual Worlds

The earliest open-ended virtual worlds were text-based, object-oriented, multiuser dungeons (MOOs).<sup>1</sup> MOOs are persistent, multiuser, interactive systems that can be thought of as low-bandwidth virtual worlds. MOOs such as *LambdaMOO* [8] are distinguished from multiuser dungeons (MUDs) by the ability for users to perform object-oriented programming within the MOO server, expanding and changing how the MOO server behaves to all users. Examples of such changes include authoring new rooms and objects, and changing the way the MOO interface operates. These changes are made using a MOO programming language that often features libraries of verbs that can be used by programmers in their coding.

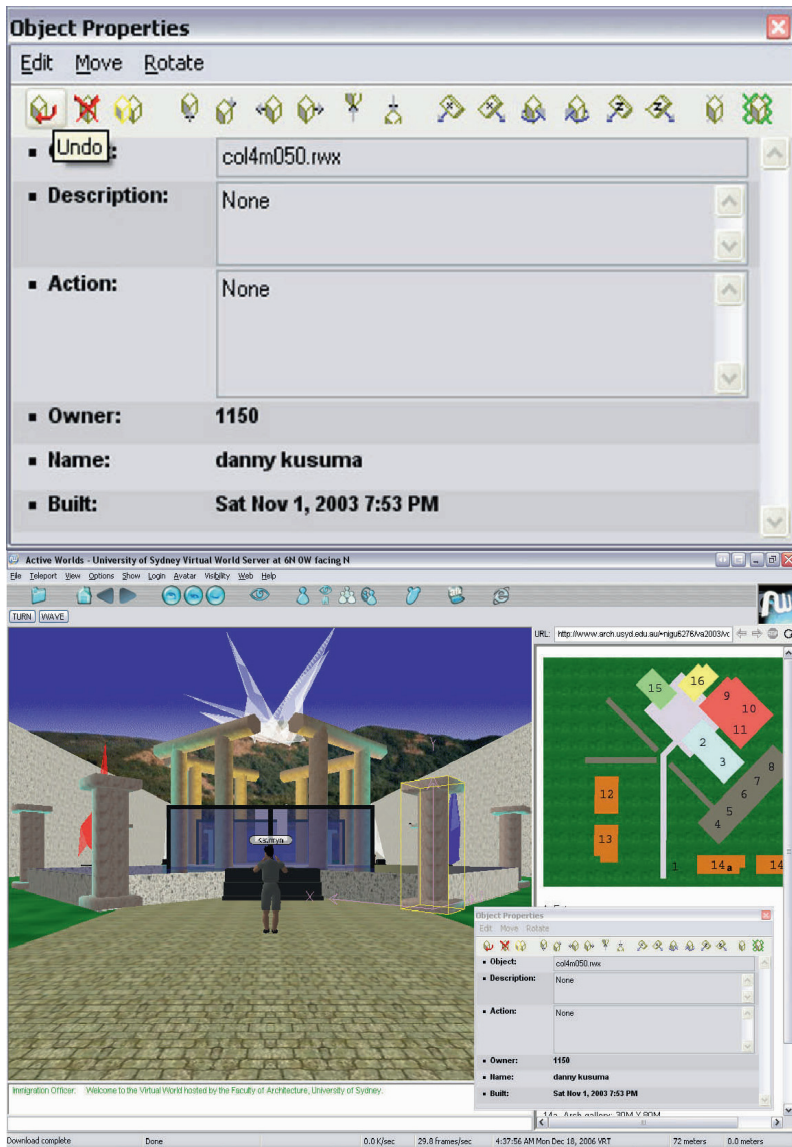
More recently, the improvement in computer graphics technology has made large-scale, 3D virtual worlds possible. Screen shots from *Second Life* and *Active Worlds*, including their 3D modelling tools, are shown in Fig. 1.1 and Fig. 1.2. In these worlds, the virtual landscape can be extended using a combination of primitive shapes and textures to create new buildings, plants, animals and other artefacts. These artefacts can also be assigned dynamic behaviours using in-world scripting tools. The ability to define object

---

<sup>1</sup> MOO = MUD, Object-Oriented.



**Fig. 1.1** In the *Second Life* virtual world, complex designs can be created using a combination of primitives, uploaded textures and scripts. Image from [9].



**Fig. 1.2** In *Active Worlds*, complex designs can be created using basic shapes, textures and actions. Image from [9].

geometry and assign natural language labels, descriptions and behaviours to designed objects are key aspects that distinguish these virtual worlds as open-ended environments.

While the users of MOOs and 3D virtual worlds have used their open-ended expansion capacities to develop games within these environments, these games have tended to adhere to existing genres such as adventure and role-playing games. The development of games that incorporate NPCs that can respond to the open-ended building capabilities of virtual worlds has been difficult due to the lack of artificial intelligence techniques that can operate in such environments.

## 1.2 Character Roles in Multiuser Games

NPCs fall into three main categories: enemies, partners and support characters [3]. Enemies are characters that oppose human players in a pseudo-physical sense by attacking the virtual life force of the human player with weapons or magic. Enemies and competitors form a critical character group in a wide range of game genres from action games to strategy or sport games. As a result, the artificial intelligence techniques for enemies and competitors have been a focus of game development.

Partner characters may perform a number of roles within a game. From a competitive perspective, partners take the opposite role to enemies and attempt to protect human players with whom they are allied. Alternatively, however, partner characters might perform non-combat roles to support their human ally. Vendor characters in *Ultima Online*, for example, sell goods on behalf of their human ally. In some games, partner characters may also be taught to perform certain behaviours by players. As with enemy characters, the combat role of partner characters has been well developed. Similarly, existing learning algorithms such as neural networks and reinforcement learning have been used successfully to achieve partner characters that learn from human supervision. The development of characters in a supporting role, however, has so far received less attention.

Support characters are the merchants, tradesmen, guards, innkeepers and so on who support the storyline of the game by offering quests, advice, goods for sale or training. Support characters expand the requirements of artificial intelligence in games beyond those of tactical enemies to social dialogue and behaviours. However, in contrast to enemies or competitors, support characters are among the least sophisticated artificially intelligent characters in current computer games, limited in both their interactions with players and in their behaviour [3]. Conversations with players tend to take the form of scripted speeches interspersed with decision points at which players can affect the direction of the conversation. The behaviour of support characters is generally

limited to a looping animation with a few scripted action sequences triggered by a player's exploits. As a result, support characters tend to stand in one place, continually chopping wood or drinking from their mug of ale, until a player initiates an interaction. During the scripted interaction the support character may gesture or move a short distance. However once the interaction is over the character returns to its looping behaviour. The following sections describe the existing artificial intelligence approaches that achieve this behaviour in further detail and highlight the need for new, more adaptive, character control algorithms.

### **1.3 Existing Artificial Intelligence Techniques for Non-Player Characters in Multiuser Games**

Existing technologies used to create NPCs in MMORPGs fall into two broad categories: reflexive agents and learning agents. Reflexive approaches, such as state machines and rule based algorithms, have been common in enemy and support characters while learning algorithms have been used in partners and some enemy characters.

#### ***1.3.1 Reflexive Agents***

Reflexive behaviour [10] is a preprogrammed response to the state of the environment – a reflex without reasoning. Only recognised states will produce a response. NPCs such as enemies, partners and support characters commonly use reflexive techniques such as state machines, fuzzy logic and rule-based approaches to define their behaviour.

#### **Rule-Based Approaches**

Rule-based approaches define a set of rules about states of the game world of the form: if <condition> then <action>. If the NPC observes a state that fulfils the <condition> of a rule, then the corresponding <action> is taken. Only states of the world that meet a <condition> will produce an <action> response. An example rule from a warrior NPC in the *Baldur's Gate* role-playing game (RPG) [11] is shown in Fig. 1.3. The condition component of this rule is an example of how such rules are domain-dependent as it makes the assumption that the character's environment contains enemies.

```

IF
    !Range(NearestEnemyOf(Myself),3)
    Range(NearestEnemyOf(Myself),8)
THEN
    RESPONSE #40
    EquipMostDamagingMelee()
    AttackReevalutate(NearestEnemyOf(Myself),60)
    RESPONSE #80
    EquipRanged()
    AttackReevalutate(NearestEnemyOf(Myself),30)
END

```

**Fig. 1.3** An example rule from a warrior character in *Baldur's Gate* [11].

```

startup state Startup${
    trigger OnGoHandleMessage$(WE ENTERED WORLD){
        if (godb.IsEditMode()){
            SetState Finish$;
        }
        else{
            SetState Spawn$;
        }
    }
}

state Spawn${
    event OnEnterState${
        GoCloneReq cloneReq$ = MakeGoCloneReq(
            "gpg gremal caged" );
        cloneReq$.StartingPos = owner.Go.Placement.Position;
        cloneReq$.SetStartingOrient =
            owner.Go.Placement.Orientation;
        cloneReq$.SnapToTerrain = true;
        newGoid$ = GoDb.SCloneGo( cloneReq$ );
    }
    event OnGoHandleMessage$(eWorldEvent e$, WorldMessage msg${
        ...
        Goid Master$;
        Master$ = msg$.GetSendFrom();
        if(master$.Go.Actor.GetSkillLevel("Combat Magic") > 0.01){
            Report.SScreen(master$.Go.Player.MachineId,
                Report.Translate(owner.go.getmessage("too evil")));
        }
        else{
            ...
        }
    }
}

state Finish${
}

```

**Fig. 1.4** An example of part of a state machine for a *Dungeon Siege* Gremel [12].



## State Machines

State machines can be used to divide a NPC's reasoning process into a set of internal states and transitions. In the *Dungeon Siege* RPG, for example, each state contains a number of `event` constructs that cause actions to be taken based on the state of the game world. Triggers define when the NPC should transition to another internal state. An example of part of a state machine for a beast called a 'Gremel' [12] is shown in Fig. 1.4. As only characters that multiply would require a spawn state, this example shows how the states are character-dependent. In addition, the condition components of the rules within the states are again heavily domain-dependent – assuming for example that the environment contains characters that have a combat magic attribute.

## Fuzzy Logic

Fuzzy logic provides a way to infer a conclusion based on facts that may be vague, ambiguous, inaccurate or incomplete [2]. *Close Combat 2* is an example of a game using fuzzy logic. Fuzzy logic uses fuzzy rules of the form `if <X is A> then <Y is B>`. X and Y are linguistic variables representing characteristics being measured – such as temperature, speed or height – while A and B represent fuzzy categories – such as hot, fast or tall. Fuzzy categories define decision thresholds within which certain courses of action may be pursued. Fuzzy logic can be applied to both rule-based approaches and state machines. While fuzzy logic allows characters to reason in environments where there is uncertainty, the ability of characters to adapt is still limited by the set of predefined fuzzy rules.

An extension of fuzzy logic is fuzzy state machines used in multiplayer simulation games such as *The Sims*. Fuzzy state machines combine state machine and fuzzy logic technologies to create agents that can identify and respond to states that approximately meet some predefined conditions [13]. For example, where simulation game NPCs controlled by state machines may consider balls as a target for kicking, NPCs controlled by fuzzy state machines may consider any object that, within some threshold, fits the description of 'being round' as a target for kicking. Characters with different personalities can be defined by building fuzzy state machines with different decision thresholds.

While fuzzy state machines have been used with success in existing simulation games, the need to define states and thresholds before the character is introduced to its environment limits the character to action within the predefined boundaries of states and decision thresholds. This technology thus becomes problematic in environments that can be modified in an open-ended manner.

In a departure from purely reflexive techniques, the support characters in some RPGs, such as *Blade Runner*, have simple goals. However these have also tended to be fairly narrow, supported by only a limited set of behaviours. Motivated reinforcement learning (MRL) offers an alternative to the design of fixed rule sets, states or goals in which a single agent model can be used to achieve multiple different characters, based on their experiences in their environment.

## **Flocking**

Flocking [14] is a special example of rule-based reasoning used to control groups of characters such as crowds or animals. Flocking uses three rules governing the separation, alignment and cohesion of individuals in a flock, herd or other kind of group. Separation rules steer an individual to avoid others, alignment rules steer an individual towards the average heading of the flock and cohesion rules steer an individual towards the average position of the flock. Flocking algorithms have been used with great success to represent lifelike crowd and animal movement and have been incorporated in games such as *Half-Life* and *Unreal*. Using the basic flocking rules, flocks can adapt to changes in their environment by moving around, towards or away from objects. However, flocking does not allow character individuality or more complex adaptation.

### **1.3.2 Learning Agents**

Learning agents are able to modify their internal structure in order to improve their performance with respect to some task [15]. In some games such as *Black and White*, NPCs can be trained to learn behaviours specified by their human master. The human provides the NPC with a reward such as food or patting to encourage desirable behaviour and punishment to discourage unwanted actions. While the behaviour of these characters may potentially evolve in any direction desired by the human, behaviour development relies on reward from human players, making it inappropriate for characters such as enemies or support characters. Learning algorithms used in games include decision trees, neural networks and reinforcement learning.

## **Decision Trees**

Decision trees are hierarchical graphs learned from a training set of previously made decisions [16]. Internal nodes in the tree represent conditions about

states of the environment, while leaf nodes represent actions. If all conditions on the path to a leaf node are fulfilled, the corresponding action can be taken. In *Black and White*, for example, creatures can learn decision trees about what food to eat based on how tasty the creature finds previously eaten food provided by a human player. While decision trees allow characters to learn, thus permitting more adaptable characters than reflexive approaches, they require a set of examples from which to learn. These examples must be provided by players. While this is appropriate for partner characters, it is generally inappropriate for enemies and support characters to have their behaviour determined only by players.

## Neural Networks

Artificial neural networks comprise a network of computational neurons with interconnecting pathways [16]. Neural networks, like decision trees, learn from examples. Examples of correct actions in different situations are fed into the network to train a character. When a character encounters a similar situation it can make a decision about the correct action to take based on the data stored in the neural network. Neural networks are used by characters in games such as *Battlecruiser: 3000 AD*, but, in many cases, the neural network is frozen before the release of a game to prevent further learning during the game. Further learning from character actions can produce networks that adapt erratically or unpredictably to players' actions.

## Reinforcement Learning

Researchers from *Microsoft* have shown that it is possible to use reinforcement learning (RL) [17] to allow NPCs to develop a single skill by applying it to fighting characters for the *Xbox* game, *Tao Feng* [18]. RL agents learn from trial-and-error and reward. After each interaction with its environment, a RL agent receives an input that contains some indication of the current state of the environment and the value of that state to the agent. This value is called a reward signal. The agent records the reward signal by updating a behavioural policy that represents information about the reward received in each state sensed so far. The agent then chooses an action that attempts to maximise the long-run sum of the values of the reward signal. In *Tao Feng*, while NPCs using RL can adapt their fighting techniques over time, it is not possible for them to identify new skills to learn about as they are limited by a pre-programmed reward for fighting. MRL offers an alternative approach for the design of learning characters that overcomes this limitation.

### 1.3.3 *Evolutionary Agents*

Evolutionary approaches such as genetic algorithms [19] simulate the process of biological evolution by implementing concepts such as natural selection, reproduction and mutation. Individuals in a population are defined in terms of a digital chromosome. When individuals reproduce, offspring are defined by a combination of their parent's chromosomes via processes of crossover and mutation. Offspring are then evaluated using a fitness function to determine which will remain in the population and which will be removed (die). Evolutionary algorithms are robust search methods that can optimise complex fitness functions. However, when genetic algorithms are used in NPCs, fitness functions must be predefined by game designers. As in RL, the fitness function limits the adaptability of a given population of individuals to the skills or tasks defined by the fitness function.

### 1.3.4 *Smart Terrain*

A key paradigm to arise from simulation games is the smart terrain concept developed by Will Wright for *The Sims* [11]. Smart terrain discards the character-oriented approach to reasoning using artificial intelligence and embeds the behaviours and possible actions associated with a virtual object within the object itself. For example, the file for the model of a television in *The Sims* might contain the instructions for watching it, turning it on and off, the conditions under which a 'Sim' might want to watch it and how a Sim should be animated while watching it. This approach allows new objects to be inserted into the game at any point, either as an expansion pack by game designers or using content creation tools by players. Achieving character adaptability using this approach, however, requires character behaviours to be explicitly programmed in each new object. This requires development effort from game designers and, while compelling for some gamers, is not interesting for others.

Expansion packs and content creation tools in general are approaches by which game designers have attempted to extend the lifetime of games by extending or allowing players to extend the original game through the addition of new content. Games in which open-ended modification of the game world is allowed while the game is in progress have the potential for a longer lifetime though the provision of more open-ended game play.

## 1.4 Summary

Multiuser games and open-ended virtual worlds are starting to be used for a broad range of activities, going beyond entertainment, which can be enhanced and supported by believable and adaptable NPCs. The in-world object modelling and programming capacity of virtual worlds such as *Second Life* provides a way for players to create and modify both the structure of virtual terrain, and the geometry, media content, and behaviour of world artefacts. This significantly changes players' expectations for believable NPCs and there is now a need for NPCs that can adapt to open-ended changes to their environment. In future the roles undertaken by NPCs in virtual worlds may also expand to encompass facilitators or arbitrators as well as traditional roles such as enemies and partners. This creates a need for new kinds of character control technology to enable NPCs to be capable of these more complex roles.

Current artificial intelligence approaches to developing the behaviour of NPCs include preprogrammed reflexive behaviours using techniques such as rules and state machines, and learned behaviours using techniques such as neural networks and reinforcement learning. The approaches have limitations in open-ended virtual worlds because they require specific knowledge embedded in the code about the state of the world and goals of the NPC. Using these techniques as a starting point, new learning approaches that include motivation as a trigger have the potential to create a new kind of NPC that is curious about the changes in the environment and is self-motivated to learn more about the changes.

This book introduces MRL as a new technique that transforms the design of NPCs by providing individual NPCs with a mechanism for open-ended, online adaptation of their own behaviour. The aim of this book is to provide game programmers with the knowledge required to develop adaptable, intelligent NPCs that can take on a life of their own in dynamic, multiuser virtual worlds. We begin in the next chapter by examining human motivation with a view to understanding how it can be embodied in artificial agents to achieve self-motivated, adaptive NPCs.

## 1.5 References

- [1] R. Bartle, *Designing virtual worlds*, New Riders, Indianapolis, 2004.
- [2] P. Baillie-de Byl, *Programming believable characters for computer games*, Charles River Media, Hingham, Massachusetts, 2004.
- [3] J. Laird and M. van Lent, Human-level AI's killer application: interactive computer games. *AI Magazine*, pp. 15–25, Summer 2001.

- [4] D. Zeltzer, Autonomy, interaction and presence. Presence: Teleoperators and Virtual Environments 1(1):127–132, 1992.
- [5] Linden, Second Life, [www.secondlife.com](http://www.secondlife.com) (Accessed January, 2007).
- [6] Active Worlds, [www.activeworlds.com](http://www.activeworlds.com) (Accessed January, 2007).
- [7] There.com [www.there.com](http://www.there.com) (Accessed July, 2008).
- [8] F. Rex, LambdaMOO: An introduction, <http://www.lambdamoo.info> (Accessed December, 2006).
- [9] K. Merrick and M.L. Maher, Motivated reinforcement learning for adaptive characters in open-ended simulation games, ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2007), ACM, Salzburg, Austria, pp. 127–134, 2007.
- [10] M.L. Maher and J.S. Gero, Agent models of 3D virtual worlds, ACADIA 2002: Thresholds, California State Polytechnic University, Pomona, pp. 127–138, 2002.
- [11] S. Woodcock, Games making interesting use of artificial intelligence techniques. <http://www.gameai.com/games.html> (Accessed October, 2005).
- [12] Siege University, 303 Skrit, <http://garage.gaspowered.com> (Accessed March, 2006).
- [13] D. Johnson and J. Wiles, Computer games with intelligence, The Tenth IEEE International Conference on Fuzzy Systems, pp. 1355–1358, 2001.
- [14] C. Reynolds, Flocks, herds and schools: a distributed behavioural model. Computer Graphics 21(4):25–34, 1987.
- [15] N.J. Nilsson, Introduction to machine learning, <http://ai.stanford.edu/people/nilsson/mlbook.html> (Accessed January, 2006), 1996.
- [16] S.J. Russell and P. Norvig, Artificial intelligence: a modern approach, Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [17] R.S. Sutton and A.G. Barto, Reinforcement learning: an introduction, The MIT Press Cambridge, Massachusetts, London, England, 2000.
- [18] T. Graepel, R. Herbrich and J. Gold, Learning to fight, The International Conference on Computer Games: Artificial Intelligence, Design and Education, 2004.
- [19] D. Goldberg, Genetic algorithms, Addison-Wesley, Reading, Massachusetts, 1989.

Motivated Reinforcement Learning

Curious Characters for Multiuser Games

Merrick, K.E.; Maher, M.L.

2009, XIV, 206 p. 118 illus., 32 illus. in color., Hardcover

ISBN: 978-3-540-89186-4